



Atty. Dkt. No. 043034-0180

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicant: **Takashi TAKENAKA**

Title: **LOGIC VERIFICATION AND LOGIC CONE EXTRACTION  
TECHNIQUE**

Appl. No.: **10/612,193**

Filing Date: **07/03/2003**

Examiner: **Unassigned**

Art Unit: **Unassigned**

**CLAIM FOR CONVENTION PRIORITY**

Commissioner for Patents  
PO Box 1450  
Alexandria, Virginia 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application:

Japanese Patent Application No. 2002-195514  
filed 07/04/2002.

Respectfully submitted,

Date: August 20, 2003

FOLEY & LARDNER  
Customer Number: 22428



**22428**

PATENT TRADEMARK OFFICE

Telephone: (202) 672-5407  
Facsimile: (202) 672-5399

By

for /

*Philip J. Anticola* <sup>Reg. No.  
38,818</sup>  
David A. Blumenthal  
Attorney for Applicant  
Registration No. 26,257

日本国特許庁  
JAPAN PATENT OFFICE

US

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2002年 7月 4日

出願番号

Application Number:

特願2002-195514

[ ST.10/C ]:

[ J P 2002-195514 ]

出願人

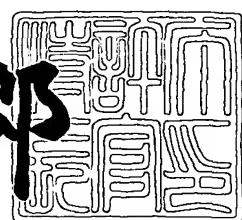
Applicant(s):

日本電気株式会社

2003年 5月 6日

特許庁長官  
Commissioner,  
Japan Patent Office

太田信一郎



出証番号 出証特2003-3032897

【書類名】 特許願  
【整理番号】 34403168  
【提出日】 平成14年 7月 4日  
【あて先】 特許庁長官殿  
【国際特許分類】 G06F 17/50  
【発明者】  
【住所又は居所】 東京都港区芝五丁目 7番1号 日本電気株式会社内  
【氏名】 竹中 崇  
【特許出願人】  
【識別番号】 000004237  
【氏名又は名称】 日本電気株式会社  
【代理人】  
【識別番号】 100088959  
【弁理士】  
【氏名又は名称】 境 廣巳  
【手数料の表示】  
【予納台帳番号】 009715  
【納付金額】 21,000円  
【提出物件の目録】  
【物件名】 明細書 1  
【物件名】 図面 1  
【物件名】 要約書 1  
【包括委任状番号】 9002136  
【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 論理検証システム及び方法、論理コーン抽出装置及び方法、論理検証及び論理コーン抽出プログラム

【特許請求の範囲】

【請求項1】 プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出手段を備えたことを特徴とする論理検証システム。

【請求項2】 RTレベル記述から論理コーンを抽出する第2の論理コーン抽出手段と、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コーンとの等価性を検証する論理コーン比較手段とを備えたことを特徴とする請求項1記載の論理検証システム。

【請求項3】 前記第1の論理コーン抽出手段は、シンボルシミュレーション手段を含むことを特徴とする請求項1または2記載の論理検証システム。

【請求項4】 プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を記憶する記憶手段と、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出手段と、

前記対応情報で指定される前記RTレベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出手段と、

前記対応情報で指定される前記動作レベル記述と前記R T レベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コーンとを比較する論理コーン比較手段とを備えたことを特徴とする論理検証システム。

【請求項5】 プログラム言語で記述された動作レベル記述をコンパイルして得たC P Uで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出手段と、前記動作レベル記述が満たすべき性質であるプロパティを記憶する手段と、前記第1の論理コーン抽出手段で抽出された論理コーンとともに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査するモデル検査手段とを備えたことを特徴とする論理検証システム。

【請求項6】 プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を入力する入力手段と、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーション手段と、

前記シンボルシミュレーション手段で得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力手段とを含むことを特徴とする論理コーン抽出装置。

【請求項7】 プログラム言語で記述された動作レベル記述をコンパイルして得たC P Uで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップを含むことを特徴とする論理検証方法。

【請求項8】 R T レベル記述から論理コーンを抽出する第2の論理コーン抽出ステップと、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンとの等価性を検証する論理コーン比較ステップとを含むことを特徴とする請求項7記載の論理検証方

法。

【請求項9】 前記第1の論理コーン抽出ステップは、シンボルシミュレーションを含むことを特徴とする請求項7または8記載の論理検証方法。

【請求項10】 プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出ステップと、

前記対応情報で指定される前記RTレベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出ステップと、

前記対応情報で指定される前記動作レベル記述と前記RTレベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを含むことを特徴とする論理検証方法。

【請求項11】 プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップと、前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、前記第1の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査する検査ステップとを含むことを特徴とする論理検証方法。

【請求項12】 プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、

前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを含むことを特徴とする論理コーン抽出方法。

【請求項13】 コンピュータに、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップを実行させる論理検証プログラム。

【請求項14】 前記コンピュータに、更に、RTレベル記述から論理コーンを抽出する第2の論理コーン抽出ステップと、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンとの等価性を検証する論理コーン比較ステップとを実行させる請求項13記載の論理検証プログラム。

【請求項15】 前記第1の論理コーン抽出ステップは、シンボルシミュレーションを含むことを特徴とする請求項13または14記載の論理検証プログラム。

【請求項16】 コンピュータに、

プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応

情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を入力する入力ステップと、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出ステップと、

前記対応情報で指定される前記R T レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出ステップと、

前記対応情報で指定される前記動作レベル記述と前記R T レベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを実行させる論理検証プログラム。

【請求項17】 コンピュータに、

プログラム言語で記述された動作レベル記述をコンパイルして得たC P Uで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップと、

前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、

前記第1の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査する検査ステップとを実行させる論理検証プログラム。

【請求項18】 コンピュータに、

プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング

グ情報を含むコンパイル情報を入力する入力ステップと、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、

前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを実行させる論理コーン抽出プログラム。

#### 【発明の詳細な説明】

##### 【0001】

##### 【発明の属する技術分野】

本発明は論理回路の設計段階における論理検証に関し、特にC P Uで実行可能なプログラム言語で記述された論理回路を対象とした論理検証技術に関する。

##### 【0002】

##### 【従来の技術】

一般に、L S IもしくはV L S Iの設計にあたっては、設計を自動的に行う、もしくは設計を支援するL S I設計自動化・支援技術が採用されている。このL S I設計自動化・支援技術を採用した設計のうち、代表的なものとしては、各種E D A (electronic design automation) ツールを利用したトップダウン設計があげられる。この種のトップダウン設計は、その上流工程から、システム設計、機能設計、論理設計、レイアウト設計と大別することができる。このトップダウン設計について図19を参照しながら以下説明する。

##### 【0003】

まず、システムの仕様作成として、L S I全体をシステムと捉え、その動作を記述することが行われる。このようにして作成された記述は、動作レベル記述12と呼ばれる。この動作レベル記述12の作成は、たとえば、自然言語、もしくはV H D L、V e r i l o g - H D Lといったハードウェア記述言語H D L (hardw

are description language)、もしくは、C言語、C++言語、java言語などのプログラム言語を用いてなされる。また、SystemC、SpecCなどといった、C言語またはC++言語に回路を表現するために便利な特徴を付加したプログラム言語を持ってなされることもある。

## 【0004】

次に、動作合成フェーズにおいて、作成された動作レベル記述12がRTレベル(register transfer level)記述14に変換される。RTレベル記述14は、通常HDLもしくはプログラム言語を用いてなされる。なお、従来、このフェーズに関しては、人手により置き換えられ、HDLを用いて、直接、RTレベル記述14を作成する場合が多かった。

## 【0005】

次いで、論理合成フェーズにおいて、RTレベル記述14が、ゲートレベル記述(ゲートレベル論理回路：ネットリスト)1Dに自動変換される。このようにして生成されたネットリストをもとに、レイアウト設計が行われ、ついで、チップ設計が行われる。

## 【0006】

LSI設計では、上記のステップによってなされた設計の正当性を調べる設計検証も重要である。設計検証には、たとえば、各合成フェーズの変換が正しいかを検証する等価性検証と、各記述が設計対象を正しく表現しているかを検証する仕様検証とがある。

## 【0007】

等価検証には、従来シミュレーションによる等価検証もしくは形式的等価性検証が用いられる。

## 【0008】

シミュレーションによる等価検証は、比較対象とする2つの記述に対して、同一のテストパターンを与えてシミュレーションを行い、その結果が所望の結果と一致するかを確認するものである。シミュレーションは、RTレベル記述14のように記述がHDLで表現されているときは専用のシミュレータを用いて行われる。これに対して、動作レベル記述12のように記述がプログラム言語で表現され

ているときは、たとえばコンパイラを用いてCPUで実行可能な形式（オブジェクトコードとよばれる）に変換された後、得られたオブジェクトコード15をCPUで直接実行して行われる。

## 【0009】

形式的等価検証は、2つの組み合わせ回路の論理が等価であるか否かを数学的手法により検証するものである。一般的な、RTレベル記述14とゲートレベル記述1Dの等価性を検査する形式的等価性検証装置の構成を示すブロック図が図20に例示される。

## 【0010】

図20を参照すると、14はRTレベル記述、1Dがゲートレベル記述である。13は両レベルにおいて比較対象となる信号の対応関係を記述した対応情報である。比較対象となる信号は、プライマリ入力、プライマリ出力、レジスタ、その他設計者によって指定される内部端子である。23はRTレベル記述、ゲートレベル記述から、それぞれ論理コーンを抽出する論理コーン抽出手段である。論理コーンとは、信号の論理を決定するために必要な組み合わせ回路のことであり、ブール式で表現される。論理コーンの抽出は、たとえば、文献1：Malik,S.et.al, “Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment”, Proceedings of IEEE International Conference on Computer-Aided Design, pp.6-9, 1988, に示される方法を用いる。具体的には、注目する変数から入力方向に組み合わせ回路をたどることにより得られる。18、1Eは、それぞれ抽出された論理コーンである。24は、対応する比較対象信号の論理コーンを表現するブール式の論理的な同一性を調べる論理コーン比較手段である。RTレベル記述14とゲートレベル記述1Dで、対応するすべての比較対象信号の論理コーンが論理的に同一であるとき、2つの記述は論理的に等価であると判定し、出力装置3に出力する。

## 【0011】

図20と同様にRTレベル記述14とゲートレベル記述1Dの等価性を検査する形式的等価性検証装置については特開平8-22485号公報にも記載されている。

## 【0012】

仕様検証にも、等価検証と同様に、シミュレーションもしくは形式的手法によるプロパティ検証という技術が知られている。シミュレーションによる仕様検証は、比較対象とする記述に対して、テストパターンを与えてシミュレーションを行い、その結果を解析することにより、回路が所望の機能を有しているかを調べる。形式的手法によるプロパティ検証は、与えられた記述が望ましい性質を満足するかを、静的かつ数学的に検査する方法である。形式的プロパティ検証は、文献2: Edmund M. Clarke, et. al. "Model Checking", pp61-74, The MIT Press, 1999に参照される。一般的な、RTレベル記述14が与えられた性質を満たしていることを検証する形式的プロパティ検証装置の構成を示すブロック図が図21に例示される。

## 【0013】

図21を参照すると、14はRTレベル記述、1Cは回路が満たすべき性質であるプロパティである。プロパティには、たとえば、「回路がデッドロックを起こさない」、「リクエスト信号が入力されたら一定サイクル以内に応答を返す」といった性質が記述される。論理コーン抽出手段23は、RTレベル記述14から論理コーン18を抽出する。モデル検査手段25は、論理コーン18をもとに、回路がとりうる信号の値の組み合わせ（状態と呼ぶ）の集合を順に列挙する。回路がとりうるすべての状態の集合を列挙したところで、これらの集合が与えられたプロパティを満たしているかのチェックを行う。

## 【0014】

形式的検証によるプロパティ検証は、シミュレーションに必要なテストデータを必要とせず、また、与えられたプロパティを満たすか否かを完全に検証することが可能であるという特徴を有する。

## 【0015】

## 【発明が解決しようとする課題】

近年のLSIの高集積化・大規模化に伴い、LSIを構成するゲート数は、数百万になることも珍しくなってきており、設計した回路が意図した機能を有するかを検証する仕様検証をいかに高速に行うかということが重要になってきてい

る。

#### 【0016】

仕様検証において、従来より一般的に行われているのはシミュレーションによる仕様検証であるが、回路が大規模化するにしたがって、RTレベル記述14上のシミュレーションに多くの時間がかかるようになってきている。これに対して、動作レベルにおける仕様検証は抽象度が高く高速であるという傾向がある。また、プログラム言語で用意された豊富なライブラリを用いてシミュレーションを行うことができる。そのため、特に、図19に示したように、動作レベル記述12をプログラム言語記述で表現し、コンパイラを用いてCPUで実行可能なオブジェクトコード15に変換した後、オブジェクトコード15をCPUで直接実行して、高速なシミュレーションを実行するが多く行われている。さらに、動作合成フェーズによって、このプログラム言語で表現された動作レベル記述12からRTレベル記述14に自動で変換するということが行われている。

#### 【0017】

この合成を行うためのツールの実態は、一般に、ソフトウェアで構成される。しかししながら、それらはソフトウェアである以上必ずしもバグがないとはいはず、各合成段階においては、適切な合成がなされたか否かの検証をする必要がある。

#### 【0018】

従来、最も一般的に行われていた動作合成フェーズにおける正当性の検証は、所定のテストパターンを用いて行われるシミュレーションであったが、このシミュレーションは以下のような問題を含んでいた。

#### 【0019】

すなわち、テストパターンを用いて行われるシミュレーションを実行するためには、当然ながら、テストパターンの生成を行わなければならず、そのために時間およびコストがかかるといった問題があった。しかも、そのシミュレーションを実行することにより検証できる合成の内容は、テストパターンに依存しており、使用したテストパターンにて考慮されていないエラーを検証することはできない。従って、従来のシミュレーションによる等価検証では、満足する結果を得られない恐れもあった。さらに、動作レベルとRTレベルの二つのレベルでシミュレ

ーションを繰り返すことは、検証の高速化を目指して動作合成を導入した目的に反するものである。

#### 【0020】

また、動作合成において動作レベル記述12から生成されたRTレベル記述14の正当性を検証する他の手法としては、たとえば動作レベル記述12との形式的等価性検証によってRTレベル記述14の正当性を検証するものも研究されている。

#### 【0021】

たとえば、動作合成の検証装置の一例が、文献3：「2000年、フォーマルメソッド・イン・システムデザイン、第16号、59-91頁( *Formal Methods in System Design, No. 16, pp.59-91, 2000* )」に記載されている。

#### 【0022】

この文献3に記載された動作合成の検証装置は、動作レベル記述およびRTレベル記述、そして動作合成の副産物である対応関係から、PVSという定理証明システムのための、1)形式仕様記述、2)等価性条件(*correctness lemmas*)、3)検証スクリプトを出力する。等価性条件とは、動作レベル記述およびRTレベル記述の分岐・合流構造を含まない基本ブロック(*basic block*)毎に実行パス(状態の系列)を抽出し、実行パスの対応関係と信号の対応関係が与えられたときに、各対応するパス毎に対応する信号値の変化が同じであることを表す論理式のことである。PVSは検証スクリプトにしたがって、各パスを実行したときの各信号値の変化を表す関数を項書換えを用いて生成する。そして、対応するパスにおける対応する信号ペアに対して、その変化を表す関数が等価であることを調べる。全ての対応するパスのペア、全ての対応する信号値のペアに対して、その変化を表す関数が等価であることをしめすことにより、動作レベル記述とRTレベル記述が等価であることを検査する。

#### 【0023】

また、動作合成の検証装置の別の一例が、文献4：「1999年、インターナショナルカンファレンスオンコンピュータデザイン、458-466頁( *International Conference on Computer Design, pp. 458-466, 1999* )」に記載されている。

## 【0024】

この文献4に記載された動作合成の検証装置は、実行パスを抽出し、実行パスの対応関係と信号の対応関係が与えられたときに、各パスを実行したときの各信号値の変化を表す関数を生成して、対応するパスの対応する信号に対してこの関数が等価であることを調べるのは前出の文献3と同じであるが、抽出する実行パスが基本ブロックに制限されない。もとの動作レベル記述の繰り返し構造（ループ構造）に対して、RTレベル記述に対応する実行パスが見つかるまでループを開しながら実行パスを抽出する。また、信号値の変化を表す関数の生成は、項書換えではなく、シンボルシミュレーションという技術を用いる。

## 【0025】

従来は、上述した手法を用いて、専ら、動作レベル記述12とRTレベル記述14の等価性を検証しており、動作レベル記述12をコンパイルして得たオブジェクトコード15とRTレベル記述14の等価性の検証については重要視されていなかった。

## 【0026】

このため、動作レベル記述12をオブジェクトコード15に変換してCPU上で実行したときの実行結果と、動作レベル記述12から実際に設計して得られた回路が動作して得られる実行結果とが完全に一致することを保証することができない恐れがあった。すなわち、プログラム言語の多くはコンパイル時にリンクされるライブラリにより言語の意味づけを変更可能であるため、動作レベル記述12とそれからコンパイルされたオブジェクトコード15とが等価である保証は必ずしもない。また、動作レベル記述12をCPUで実行可能な形式に変換するコンパイラもソフトウェアで構成される以上、必ずしもバグがないとはいはず、それが原因で動作レベル記述12とオブジェクトコード15とが等価にならない場合もある。若し、動作レベル記述12とオブジェクトコード15とが等価でないとすると、動作レベル記述12から導出されたRTレベル記述14もオブジェクトコード15と等価でないことになり、オブジェクトコード15の実行によりその動作の正しさが確認されても、RTレベル記述14の動作も正しいものと判断することはできない。このため、動作合成フェーズにおいて動作確認が正しく行わ

れていない状況で次の論理合成フェーズに進んでしまう過ちを犯してしまうことになる。

【0027】

そこで、動作レベル記述12をコンパイルして得たオブジェクトコード15をCPUで実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できる論理検証システムが望まれる。

【0028】

また、動作レベル記述12において形式的プロパティ検証を行いたいという要求もある。この場合、形式的プロパティ検証はシミュレーションと補完する関係にあるため、CPUで実行される動作に対する検証、つまりオブジェクトコード15に対する検証を行えることが望ましい。

【0029】

また、これらの論理検証を論理コーンを用いて実現するためには、HDL記述やゲートレベル記述から論理コーンを抽出する従来の論理コーン抽出技術が使えないため、CPUで実行可能な形式であるオブジェクトコードから論理コーンを抽出する技術が必要である。

【0030】

【発明の目的】

本発明の目的は、動作レベル記述をコンパイルして得たオブジェクトコードと前記動作レベル記述から導出したRTレベル記述との等価性を検証し得るようにすることで、動作レベル記述をCPUで実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できる論理検証システム及び方法を提供することにある。

【0031】

また本発明の別の目的は、動作レベル記述をCPUで実行したときの動作に対する、形式的プロパティ検証システムおよび方法を提供することにある。

【0032】

また本発明の他の目的は、オブジェクトコードから論理コーンを抽出する論理コーン抽出装置および方法を提供することにある。

## 【0033】

## 【課題を解決するための手段】

本発明の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出手段を備えることを基本とする。そして、RTレベル記述との論理検証に対して本発明を適用する場合、更に、RTレベル記述から論理コーンを抽出する第2の論理コーン抽出手段と、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コーンとの等価性を検証する論理コーン比較手段とを備える。ここで、前記第1の論理コーン抽出手段は、シンボルシミュレーション手段を含んでいて良い。

## 【0034】

より具体的には、本発明の第1の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を記憶する記憶手段と、前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出手段と、前記対応情報で指定される前記RTレベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出手段と、前記対応情報で指定される前記動作レベル記述と前記RTレベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コ

ーンとを比較する論理コーン比較手段とを備えている。

#### 【0035】

また本発明の第2の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出手段と、前記動作レベル記述が満たすべき性質であるプロパティを記憶する手段と、前記第1の論理コーン抽出手段で抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査するモデル検査手段とを備えている。

#### 【0036】

また本発明の論理コーン抽出装置は、プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を入力する入力手段と、前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーション手段と、前記シンボルシミュレーション手段で得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力手段とを含んで構成される。

#### 【0037】

また本発明の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップを含むことを基本とする。そして、RTレベル記述との論理検証に対して本発明を適用する場合、更に、RTレベル記述から論理コーンを抽出する第2の論理コーン抽出ステップと、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンどうしの等価性を検証する論理コーン比較ステップとを含む

ように構成される。ここで、前記第1の論理コーン抽出ステップは、シンボルシミュレーションを含むものであって良い。

#### 【0038】

より具体的には、本発明の第1の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報を指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報を入力する入力ステップと、前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出ステップと、前記対応情報で指定される前記RTレベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出ステップと、前記対応情報で指定される前記動作レベル記述と前記RTレベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出ステップで抽出された論理コーンと前記第2の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを含んで構成される。

#### 【0039】

また本発明の第2の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出ステップと、前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、前記第1の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティ

を満たしているかどうかを検査する検査ステップとを含んで構成される。

【0040】

また本発明の論理コーン抽出方法は、プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを含んで構成される。

【0041】

【作用】

本発明の第1の論理検証システムおよび方法にあっては、C P Uで直接実行されるオブジェクトコードから、論理等価性検証に必要な論理コーンを抽出することにより、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードとR T レベル記述との等価性を検証することができる。これにより、動作レベル記述をC P Uで実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できるような論理検証が可能となる。

【0042】

本発明の第2の論理検証システムおよび方法にあっては、C P Uで直接実行されるオブジェクトコードから、論理等価性検証に必要な論理コーンを抽出することにより、C P Uで実行される動作に対する形式的プロパティ検証が可能となる。

【0043】

本発明の論理コーン抽出装置および方法にあっては、プログラム記述中の論理コ

ーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報に基づいて、前記プログラム記述をコンパイルして得たオブジェクトコードから、前記対応情報で指定した論理コーン抽出範囲毎に、前記対応情報で指定した抽出対象信号に対応する前記オブジェクトコードの変数毎の論理コーンを自動的に抽出することができる。

## 【0044】

## 【発明の実施の形態】

次に、本発明の実施の形態について図面を参照して詳細に説明する。

## 【0045】

図1を参照すると、本発明の第1の実施の形態は、情報を記憶する記憶装置1と、プログラム制御により動作するデータ処理装置2と、ディスプレイ装置や印刷装置等の出力装置3を含む。

## 【0046】

データ処理装置2は、コンパイラ21と、オブジェクトコード論理コーン抽出手段22と、HDL論理コーン抽出手段23と、論理コーン比較手段24とを含む。

## 【0047】

記憶装置1は、コンパイルライブラリ11と、プログラム言語で表現された動作レベル記述12と、対応情報13と、HDLで表現されたRTレベル記述14とをあらかじめ記憶している。

## 【0048】

動作レベル記述12は一般に人手により記述される。その記述言語は、たとえば、C言語、C++言語、java言語、SystemC、SpecCなどである。コンパイルライブラリ11は、動作レベル記述12をオブジェクトコードにコンパイルしてCPUで動作させるために必要になるライブラリであり、たとえば、SystemCなどの標準ライブラリであったり、設計者により作成されたり、コンパイラメーカにより供給される。

## 【0049】

R T レベル記述14は、たとえば、動作合成フェーズにより作成されるものであり、たとえば、動作合成システムに動作レベル記述12を与えることで得られる。あるいは、たとえば、動作レベル記述12から人手により変換されるものである。

#### 【0050】

対応情報13は、動作合成フェーズにおいて、動作レベル記述12をR T レベル記述14に変換する際に、動作合成システムもしくは、設計者によって与えられる。対応情報13は、等価性の比較の対象とする記述の組の情報と、この組毎の等価性の比較の対象となる信号の組の情報とを備えている。比較の対象とする記述の組の情報とは、動作レベル記述12の一部（もしくは全部）を特定する情報と、この動作レベル記述12の一部（もしくは全部）から合成されるR T レベル記述14の一部（もしくは全部）を特定する情報である。たとえば、動作レベル記述12が一クロックサイクルの動作を順に記載している場合は、この一クロックサイクルの動作部分のそれぞれに対して、この記述から合成されるR T レベル記述14の一部分が組の情報を備える。

#### 【0051】

この点を図18を参照して説明すると、動作レベル記述12に含まれる或る記述Aとその記述Aに対応するR T レベル記述14の記述Bとを比較の対象とする場合、対応情報13には、記述Aを特定する情報と記述Bを特定する情報との組（A : B）が設定される。また、記述Aに信号a1、a2、記述Bに信号b1、b2がそれぞれあり、a1とb1、a2とb2、をそれぞれ比較の対象とする場合、対応情報13には、記述Aと記述Bとの組の情報（A : B）に対応して、信号a1と信号b1との対応情報（a1 : b1）、信号a2と信号b2との対応情報（a2 : b2）が設定される。なお、検証の精度を確保するためには、等価性の比較の対象とする記述の組、その組の等価性の比較の対象となる信号の組、それを網羅的に対応情報13に記述しておくことが望まれる。

#### 【0052】

コンパイラ21は、コンパイルライブラリ11と動作レベル記述12からC P Uで実行可能なオブジェクトコード15およびコンパイル情報16を生成し、記憶

装置1に記憶する。コンパイル情報16は、たとえば、ELF (executable and linking format)といった、標準フォーマットで提供される。ここで、コンパイル情報16には、動作レベル記述12とオブジェクトコード15とのマッピング情報が含まれる。このマッピング情報により、動作レベル記述12のどの記述部分が、オブジェクトコード15のどのコード部分に対応しているか、動作レベル記述12のどの信号が、オブジェクトコード15のどの変数に対応しているか、が指示される。

#### 【0053】

この点を図18を参照して説明すると、動作レベル記述12の記述Aがオブジェクトコード15上のオブジェクトコード部分Cに対応する場合、コンパイル情報16には、記述Aがオブジェクトコード部分C1に対応する旨の情報(A:C)が含まれる。また、記述Aの信号a1がオブジェクトコード15上では変数c1で表現され、記述Aの信号a2がオブジェクトコード15上では変数c2で表現される場合、信号a1と変数c1の対応情報(a1:c1)、信号a2と変数c2の対応情報(a2:c2)がコンパイル情報16によって示される。

#### 【0054】

オブジェクトコード論理コーン抽出手段22は、記憶装置1に記憶されたオブジェクトコード15およびコンパイル情報16を読み込み、対応情報13で比較の対象とされた記述毎に、その記述内の比較の対象とされた信号に対応する変数毎の論理コーン(動作レベル論理コーン17)を抽出する。抽出された動作レベル論理コーン17は記憶装置1に記憶する。

#### 【0055】

この点を図18を参照して説明すると、オブジェクトコード論理コーン抽出手段22は、対応情報13を参照して、記述Aが比較の対象となる記述の1つであり、この記述A中の信号a1、a2が比較の対象となる信号であることを認識し、コンパイル情報16を参照して、記述Aに対応するオブジェクトコード部分がCであり、信号a1、a2に対応する変数がc1、c2であることを認識する。そして、オブジェクトコード部分Cに関して、後述するようにシンボルシミュレーションを行って、変数c1の論理コーンと変数c2の論理コーンとを抽出し、記

憶装置1に記憶する。

【0056】

HDL記述論理コーン抽出手段23は、記憶装置1に記憶されたRTレベル記述14を読み込み、対応情報13で比較の対象とされた信号の論理コーン（RTレベル論理コーン18）を抽出する。抽出されたRTレベル論理コーン18は記憶装置1に記憶する。

【0057】

この点を図18を参照して説明すると、HDL記述論理コーン抽出手段23は、対応情報13を参照して、記述Bが比較の対象となる記述の1つであり、この記述中の信号b1、b2が比較の対象となる信号であることを認識し、RTレベル記述14から従来と同様にして信号b1の論理コーンと信号b2の論理コーンとを抽出し、記憶装置1に記憶する。

【0058】

論理コーン比較手段24は、記憶装置1に記憶された対応情報13で示される比較の対象となる記述の組毎に、その組の記述に含まれる比較対象信号の組を選択し、記憶装置1からこの信号の論理コーン17、18をそれぞれ取り出して、双方の論理コーン17、18が論理的に等価であることを判定する。そして、判定結果を出力装置3から出力する。

【0059】

この点を図18を参照して説明すると、論理コーン比較手段24は、対応情報13を参照して、動作レベル記述12の記述AとRTレベル記述14の記述Bとの組が比較の対象となる記述の組の1つであること、その組の記述A、Bに含まれる信号a1と信号b1、信号a2と信号b2とがそれぞれ比較対象信号の組であることをそれぞれ認識する。また、コンパイル情報16を参照して、動作レベル記述12の記述Aがオブジェクトコード15のCに対応し、信号a1、a2が変数c1、c2に対応することを認識する。そして、記憶装置1から、変数c1の論理コーンと信号b1の論理コーンとを取り出し、双方の論理コーンが論理的に等価であるかどうかを判定する。また、記憶装置1から、変数c2の論理コーンと信号b2の論理コーンとを取り出し、双方の論理コーンが論理的に等価である

かどうかを判定する。

【0060】

次に、各図を参照して本実施の形態の動作について詳細に説明する。

【0061】

図2において、ステップA1では、動作レベル記述12よりコンパイルして得たオブジェクトコード15から、比較対象信号の論理コーン17を抽出する。このステップA1の詳細は後述する。ステップA2では、RTレベル記述14から、比較対象信号の論理コーン18を抽出する。論理コーンは、対応情報13に記述された「比較対象とする記述の組」毎に「比較対象とする信号の組」それぞれについて一組ずつ抽出され、これが比較単位となる。

【0062】

ステップA3では、対応する比較対象信号の論理等価性を比較する。かかるステップを比較単位の分だけ繰り返す。すべての比較対象信号について等価であることが示されたとき、オブジェクトコード15とRTレベル記述14は等価であると結論づけられる。

【0063】

図3を参照すると、ステップA1の詳細が示されている。

【0064】

ステップA11では、動作レベル記述12をコンパイラ21によってCPU上で実行可能なオブジェクトコード15にコンパイルする。なお、既にコンパイルされたオブジェクトコード15が存在する場合には、このステップA11は省略してよい。

【0065】

ステップA12では、コンパイルされたオブジェクトコード15上で、シンボルシミュレーションを行い、比較対象信号の論理コーン17を抽出する。

【0066】

図4を参照すると、ステップA12の詳細が示されている。

【0067】

ステップA121では、オブジェクトコード15上にシンボルシミュレーション

の開始点を見つける。シンボルシミュレーションの開始点は、対応情報13およびコンパイル情報16を参照して決定される。具体的には、対応情報13に記述されている「比較の対象とする記述の組」から、動作レベル記述12の一部分が指定される。コンパイル情報16からこの部分記述の開始点に対応するオブジェクトコード15上の開始点が指示され、これがシンボルシミュレーションの開始点となる。この点を図18を参照して説明すると、対応情報13から動作レベル記述12の記述Aが1つの比較対象の記述と認識され、コンパイル情報16から記述Aがオブジェクトコード部分Cに対応することが認識され、このオブジェクトコード部分Cの開始点をシンボルシミュレーションの開始点とする。なお、後述するシンボルシミュレーションの終了点は、このオブジェクトコード部分Cの終了点である。

#### 【0068】

ステップA122では、比較対象信号に初期シンボル値を対応付ける。比較対象信号は、対応情報13およびコンパイル情報16によって指示される。シンボル値とは、変数の値を、整数などの実際の値ではなく、記号などのシンボルとして表現した値である。この点を図18を参照して説明すると、対応情報13から、動作レベル記述12の記述Aに信号a1、信号a2が比較対象信号として含まれることを認識し、コンパイル情報16から信号a1、a2はオブジェクトコード部分Cでは変数c1、c2に対応することを認識し、変数c1、c2をそれぞれ比較対象信号としてそれらに初期シンボル値を対応付ける。

#### 【0069】

ステップA123では、オブジェクトコード15上の前記シンボルシミュレーション開始点から順にオブジェクトコード15をトレースしながらオブジェクトコード15の一命令ごとにその命令の定義に従ってシンボルシミュレーションを進める。

#### 【0070】

CPUで実行されるオブジェクトコード15は、メモリやレジスタといった記憶素子からの値の読み込み、これらの値をもとにした演算、演算結果の記憶素子への書き込み、条件分岐など実行順序の変更の組み合わせからなっている。演算に

使用される値にシンボル値が含まれるときはシンボル値演算をおこなう。シンボル値演算は、実際の演算を行うのではなく、演算結果を関数などによる表現式で表すものであり、この表現式もシンボル値という。このとき、演算に使用される値に即値が含まれるときは即値をシンボル値に変換してからシンボル値演算を行う。シンボル値演算の結果得られる演算結果を表すシンボル値は指定される記憶素子に記憶される。条件分岐の分岐条件がシンボル値である場合には、分岐条件シンボル値を記憶して、分岐があるものとないものそれぞれトレースを続行する。分岐条件が即値である場合には、分岐演算の結果に従い、どちらか一方のトレースを続行する。オブジェクトコード15のトレースはオブジェクトコード15上のシンボルシミュレーション終了点に達するまで続ける。

#### 【0071】

シンボルシミュレーションの終了点に到達したとき、比較対象信号に関して或るシンボル値が得られる。このシンボル値は、具体的に言うと、オブジェクトコード15をシンボルシミュレーション開始点からシンボルシミュレーション終了点まで実行したときの比較対象信号のシンボルシミュレーション終了点時点の値を、比較対象信号のシンボルシミュレーション開始点時点で設定した初期シンボル値で表現したシンボル値である。このシンボル値は、初期シンボル値とシンボル値演算からなっており、初期シンボル値を回路の入力変数、シンボル値演算を論理演算に変換すると、組み合わせ論理式となる。こうして得られた論理式を、論理コーンとする（ステップA125）。

#### 【0072】

なお、分岐により、シンボルシミュレーション終了点に到達する経路が複数ある場合には、あらかじめ記憶された分岐条件シンボル値とシンボルシミュレーション終了点に達したときのシンボル値の論理式を作成し、マルチプレクサによってひとつのシンボル値にまとめる。そして、これを同様に論理式に変換し、これを、論理コーンとする。

#### 【0073】

対応情報13中に、動作レベル記述12に関する複数の比較対象の記述が含まれる場合、各記述毎に図4の処理が実行される。

【0074】

## 【実施例】

次に、具体的な実施例を用いて本実施の形態の動作を説明する。

【0075】

図5を参照するとC言語で記述された動作レベル記述12の例が示されている。この記述では、関数addition()の部分が動作合成によって回路に合成されことが指定されている(11行目)。関数addition()は、関数が呼ばれるたびに変数in0の値が順に累積加算されたものが変数out0に格納されるように変数a,bを用いて記述されている(12~18行目)。

【0076】

図6を参照すると図5の動作レベル記述12を変換して得られるRTレベル記述14の例が示されている。動作レベル記述12中の関数addition()が、additionという名前のモジュールとして実現されている(1~13行目)。変数in0がモジュールの入力として実現されている(2行目)。変数out0がモジュールの出力として実現されている(3行目)。関数の動作が2つのレジスタRG01およびRG02を用いて実現されている(10、11行目)。

【0077】

図8を参照すると、対応情報13の一部が例示されている。対応情報13では動作レベル記述12中の変数in0、out0、a、bとRTレベル記述14上の信号in0、out0、RG01、RG02との対応関係が与えられている。また、対応情報13では、関数addition()が動作合成により合成されたこと、および、関数addition()がモジュールadditionとして実現されたことも別途与えられる。

【0078】

いま、図5の動作レベル記述12、図6のRTレベル記述14、図8の対応情報13が与えられたとする。

【0079】

動作レベル記述12をコンパイラ21によりCPUで実行可能なオブジェクトコード15に変換する(ステップA11)。

【0080】

図7を参照すると、オブジェクトコード15の一部が例示されている。`movl a`  
`ddl`はそれぞれCPUで実行される命令を表している。`movl`はメモリ空間の指定  
アドレス（第一引数）から指定レジスタ（第二引数）への値の読み込みをする命  
令である。`ddl`は指定レジスタ上（第一引数）の値とメモリ空間の指定アドレス  
（第二引数）に格納された値との加算を計算し、指定レジスタ（第一引数）に計  
算結果を格納する命令である。`in0, a, b, out0`はメモリ空間内のアドレス位置  
を表すラベルである。`%eax`はCPUに備えられたレジスタを示している。

#### 【0081】

ラベル`addition`から始まる一連の命令によって、動作レベル記述12の関数`addition()`の動作が実現されている。2行目から3行目によって変数`a`の値が変数`b`の  
値と加算され加算結果が変数`b`に格納される。4行目から5行目によって変数`in0`  
の値が変数`a`に格納される。6行目から7行目によって、変数`b`の値が変数`out0`  
にコピーされる。

#### 【0082】

図9を参照すると、コンパイル情報16の一部が例示されている。コンパイル情  
報16では、動作レベル記述12の各変数`in0, out0, a, b`が、オブジェクトコ  
ード15中では、メモリ空間上で同名のラベルが付けられたアドレスに格納され  
ることを示している。さらに、コンパイル情報16では、関数`addition()`の処理  
が、図7に例示されたオブジェクトコードの処理に対応していることも、別途指  
定されている。

#### 【0083】

次に、図7のオブジェクトコード15から論理コーン17を抽出する（ステップ  
A12）。論理コーン17の抽出では、まず、オブジェクトコード15からシン  
ボルシミュレーション開始点を見つける。シンボルシミュレーション開始点は、  
対応情報13およびコンパイル情報16で指定される。図7に例示されたオブジ  
エクトコードでは、ラベル`addition`の後の命令がシンボルシミュレーション開始  
点となる。

#### 【0084】

次に、シンボルシミュレーションの対象となる変数に初期シンボル値を設定する

。図7に例示されたオブジェクトコード15では、シンボルシミュレーションの対象となる変数は、対応情報13およびコンパイル情報16から判明する、変数a, b, in0, out0である。

#### 【0085】

図10を参照すると、初期シンボル値の設定が例示されている。変数a, b, in0, out0に対応するメモリ空間上のアドレスに対して、初期シンボル値a'、b'、in0'、out0'がそれぞれ対応付けられている。本実施例では、変数名の後にダッシュ(')をつけて実行前の変数の値を表現するものとしている。

#### 【0086】

次に、オブジェクトコード15を順にトレースしながらシンボルシミュレーションを進める（ステップA123）。

#### 【0087】

図11を参照すると、シンボルシミュレーションの様子が例示されている。開始点から2つの命令を実行した後は、変数bの値を表すシンボル値が、変数a, bの加算結果を表すシンボル値“a' + b'”になっている。また、シンボルシミュレーション終了点までシンボルシミュレーションが進んだときは、変数aの値が、変数in0の値を表すシンボル値in0'に、変数bおよびout0の値が変数a, bの加算結果を表すシンボル値“a' + b'”になっていることを示している。これらを論理コーンとする（ステップA125）。図12を参照すると抽出された論理コーンが例示されている。

#### 【0088】

次に、RTレベル記述14から論理コーン18を抽出する（ステップA2）。図13を参照するとRTレベル記述14から抽出された論理コーン18が例示されている。

#### 【0089】

最後に、対応する信号のペアに対して抽出された論理コーン17、18が等価であるかを調べる（ステップA3）。図8および図9を参照すると、対応する信号のペアは図14に示される。すべてのペアにおいて論理コーン17、18が等価である場合は、オブジェクトコード15とRTレベル記述14とが等価であると

結論づける。

【0090】

【発明の第2の実施の形態】

本発明の第2の実施の形態を、図面を用いて詳細に説明する。

【0091】

本発明の第2の実施の形態では、RTレベル記述がHDLで与えられる代わりに、プログラム言語で与えられる点が異なる。

【0092】

図15を参照すると、本発明の第2の実施の形態は、RTレベル記述14からのRTレベル論理コーン18の抽出を、図1に示されるHDL論理コーン抽出手段23のかわりに、コンパイラ21A、オブジェクトコード論理コーン抽出手段22Aで行なうことが異なる。

【0093】

コンパイラ21Aは、コンパイルライブラリ11Aとプログラム言語で与えられたRTレベル記述14からCPUで実行可能なオブジェクトコード1Aおよび、RTレベル記述14とオブジェクトコード1Aとの間の、変数とプログラムの格納位置の対応関係を記したコンパイル情報1Bを生成し、記憶装置1に記憶する。

【0094】

オブジェクトコード論理コーン抽出手段22Aは、記憶装置1に記憶されたオブジェクトコード1Aを読み込み、対応情報13とコンパイル情報1Bを参照しながら、比較の対象とされた記述毎に、その記述に含まれる変数の論理コーン（動作レベル論理コーン17）を抽出する。その動作は、オブジェクトコード論理コーン抽出手段22と基本的に同じである。

【0095】

論理コーン比較手段24が動作レベル論理コーン17とRTレベル論理コーン18の等価性を比較することは、本発明の第1の実施の形態と同一である。

【0096】

次に、本実施の形態の効果について説明する。

【0097】

プログラム言語を用いてRTレベル記述をおこなうことで、プログラム言語で用意された豊富なライブラリを用いてシミュレーションを行うことができる。その場合、動作合成システムはHDLではなくプログラム言語をもちいてRTレベル記述を出力するという要求も増えてきている。本実施の形態では、このように、RTレベル記述としてHDL記述ではなく、プログラム記述が与えられても、与えられた二つの記述の等価性を検証することができる。

【0098】

【発明の第3の実施の形態】

本発明の第3の実施の形態を、図面を用いて詳細に説明する。

【0099】

本発明の第3の実施の形態では、2つの記述があたえられて、それらの等価性を検証するのではなく、与えられたプログラム記述が与えられたプロパティを満たしていることを証明する点が異なる。

【0100】

図16を参照すると、本発明の第3の実施の形態は、モデル検査手段25を備え、形式的等価検証ではなく、形式的プロパティ検証を行う点で異なる。

【0101】

コンパイラ21、オブジェクトコード論理コーン抽出手段22を備え、動作レベル記述12、コンパイルライブラリ11、対応情報13から、動作レベル論理コーン17を抽出する点は、本発明の第一の実施の形態と同一である。

【0102】

モデル検査手段25は、動作レベル論理コーン17と、プロパティ1Cがあたえられ、論理コーン17をもとに、回路がとりうる信号の値の組み合わせ（状態と呼ぶ）の集合を順に列挙する。回路がとりうるすべての状態の集合を列挙したところで、これらの集合が与えられたプロパティを満たしているかのチェックを行う。

【0103】

図17を参照すると、本発明の第3の実施の形態の動作が示されている。

## 【0104】

ステップA1では、プログラム言語で記述された動作レベル記述12を、コンパイラ21でCPUが実行可能なオブジェクトコード15にコンパイルする。

## 【0105】

ステップA2では、コンパイルされたオブジェクトコード15から、論理コーン17を抽出する。

## 【0106】

ステップA3では、モデル検査手段25によって、論理コーン17をもとに、動作レベル記述12が与えられた性質を満足するかどうかを判定し、判定結果を出力装置3に出力する。モデル検査手段25には、たとえば前出の既存の技術を利用する。

## 【0107】

次に、本実施の形態の効果について説明する。本実施の形態では、RTレベル記述の代わりに、プログラム言語で記述された動作レベル記述12が満たすべきプロパティ1Cが与えられたときでも、動作レベル記述12がプロパティを満たすことを検証できる。

## 【0108】

以上本発明の実施の形態および実施例について説明したが、本発明は以上の実施の形態および実施例にのみ限定されるものでなく、その他各種の付加変更が可能である。また、本発明の論理検証システム及び論理コーン抽出装置は、その有する機能をハードウェア的に実現することは勿論、コンピュータと論理検証プログラム、論理コーン抽出プログラムとで実現することができる。論理検証プログラム、論理コーン抽出プログラムは、磁気ディスクや半導体メモリ等のコンピュータ可読記録媒体に記録されて提供され、コンピュータの立ち上げ時などにコンピュータに読み取られ、そのコンピュータの動作を制御することにより、そのコンピュータを前述した各実施の形態における論理検証システム及び論理コーン抽出装置として機能させる。

## 【0109】

## 【発明の効果】

以上説明したように本発明の第1の論理検証システムおよび方法によれば、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードとRTレベル記述との等価性を検証することができる。これにより、動作レベル記述をCPUで実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できるような論理検証が可能となる。

#### 【0110】

また本発明の第2の論理検証システムおよび方法によれば、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードに対する形式的プロパティ検証が可能となる。

#### 【0111】

また本発明の論理コーン抽出装置および方法によれば、対応情報とコンパイル情報に基づいて、プログラム記述をコンパイルして得たオブジェクトコードから、前記対応情報で指定した論理コーン抽出範囲毎に、前記対応情報で指定した抽出対象信号に対応する前記オブジェクトコードの変数毎の論理コーンを自動的に抽出することができる。

#### 【図面の簡単な説明】

##### 【図1】

本発明の第1の実施の形態の構成を示すブロック図である。

##### 【図2】

第1の実施の形態の動作を示す流れ図である。

##### 【図3】

第1の実施の形態の動作を示す流れ図である。

##### 【図4】

第1の実施の形態の動作を示す流れ図である。

##### 【図5】

第1の実施の形態の動作の具体例を示す図である。

##### 【図6】

第1の実施の形態の動作の具体例を示す図である。

【図7】

第1の実施の形態の動作の具体例を示す図である。

【図8】

第1の実施の形態の動作の具体例を示す図である。

【図9】

第1の実施の形態の動作の具体例を示す図である。

【図10】

第1の実施の形態の動作の具体例を示す図である。

【図11】

第1の実施の形態の動作の具体例を示す図である。

【図12】

第1の実施の形態の動作の具体例を示す図である。

【図13】

第1の実施の形態の動作の具体例を示す図である。

【図14】

第1の実施の形態の動作の具体例を示す図である。

【図15】

本発明の第2の実施の形態の構成を示すブロック図である。

【図16】

本発明の第3の実施の形態の構成を示すブロック図である。

【図17】

第3の実施の形態の動作を示す流れ図である。

【図18】

本発明の第1の実施の形態の動作説明図である。

【図19】

トップダウン設計の流れの概略を示す図である。

【図20】

従来の形式的等価性検証装置のブロック図である。

【図21】

従来の形式的プロパティ検証装置のブロック図である。

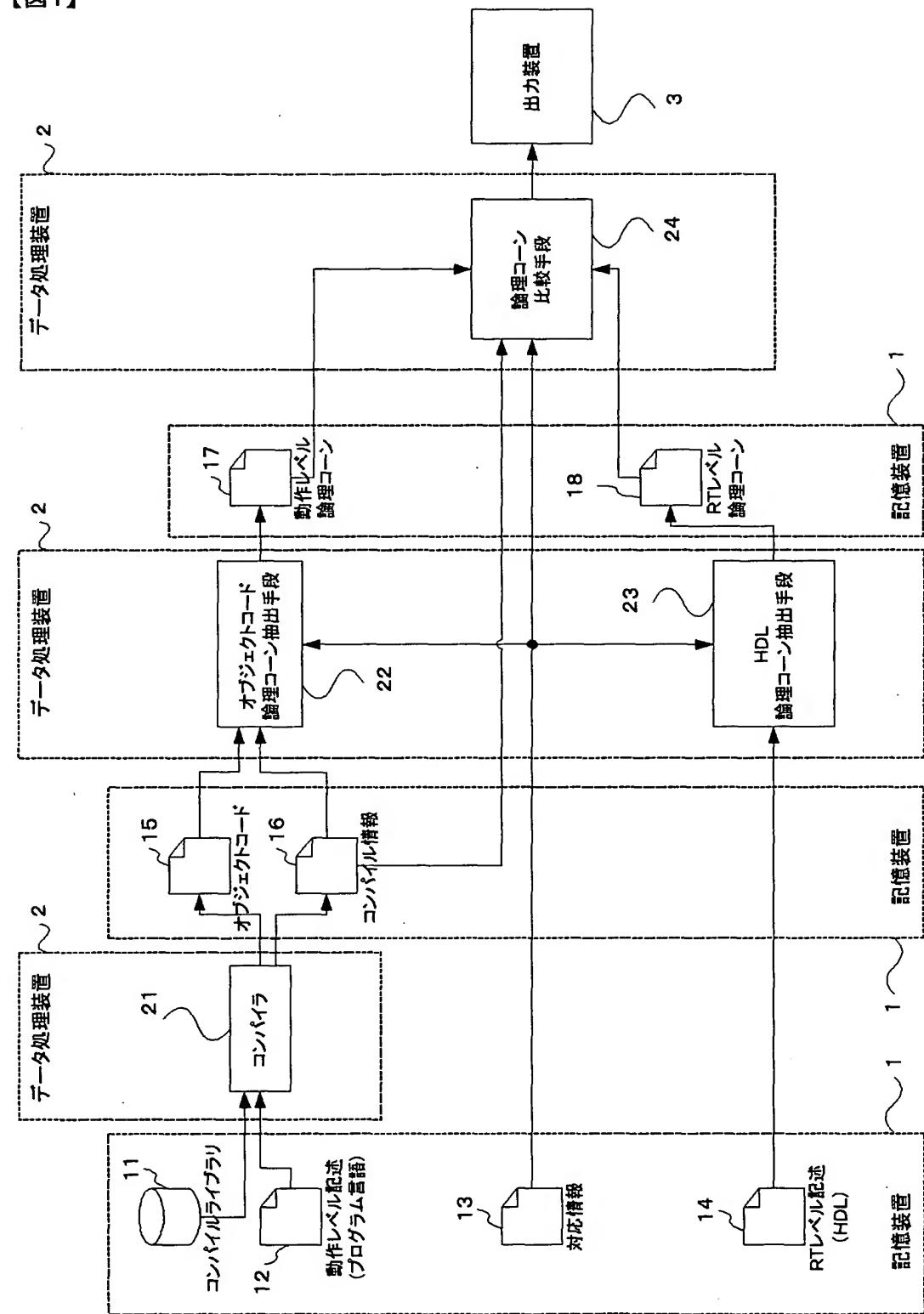
【符号の説明】

- 1 …記憶装置
- 2 …データ処理装置
- 3 …出力装置
- 1 1、 1 1 A …コンパイルライブラリ
- 1 2 …動作レベル記述
- 1 3 …対応情報
- 1 4 …R T レベル記述
- 1 5、 1 A …オブジェクトコード
- 1 6、 1 B …コンパイル情報
- 1 7 …動作レベル論理コーン
- 1 8 …R T レベル論理コーン
- 1 C …プロパティ
- 1 D …ゲートレベル記述
- 1 E …ゲートレベル論理コーン
- 2 1、 2 1 A …コンパイラ
- 2 2、 2 2 A …オブジェクトコード論理コーン抽出手段
- 2 3 …H D L 論理コーン抽出手段
- 2 4 …論理コーン比較手段

【書類名】 図面

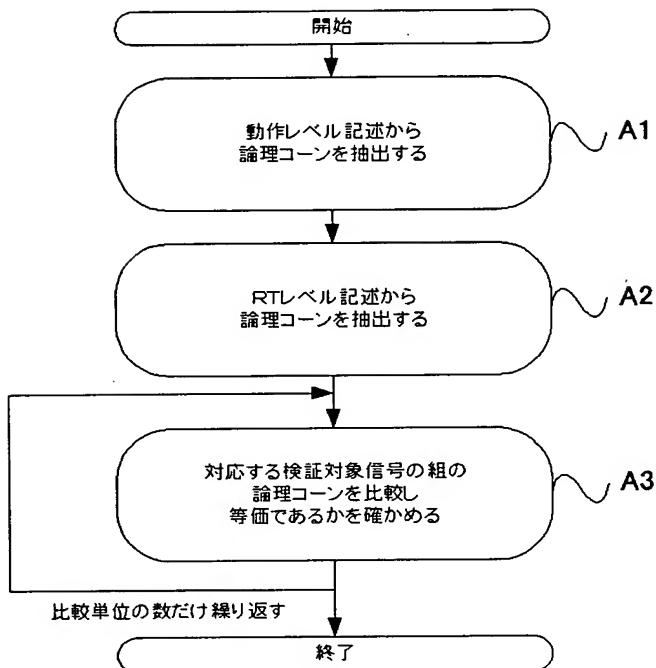
【図1】

【図1】



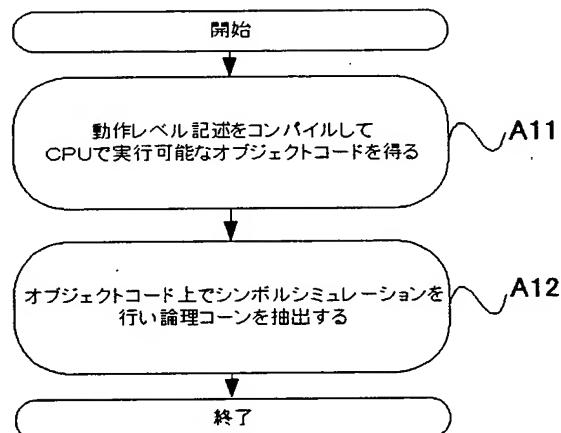
【図2】

【図2】



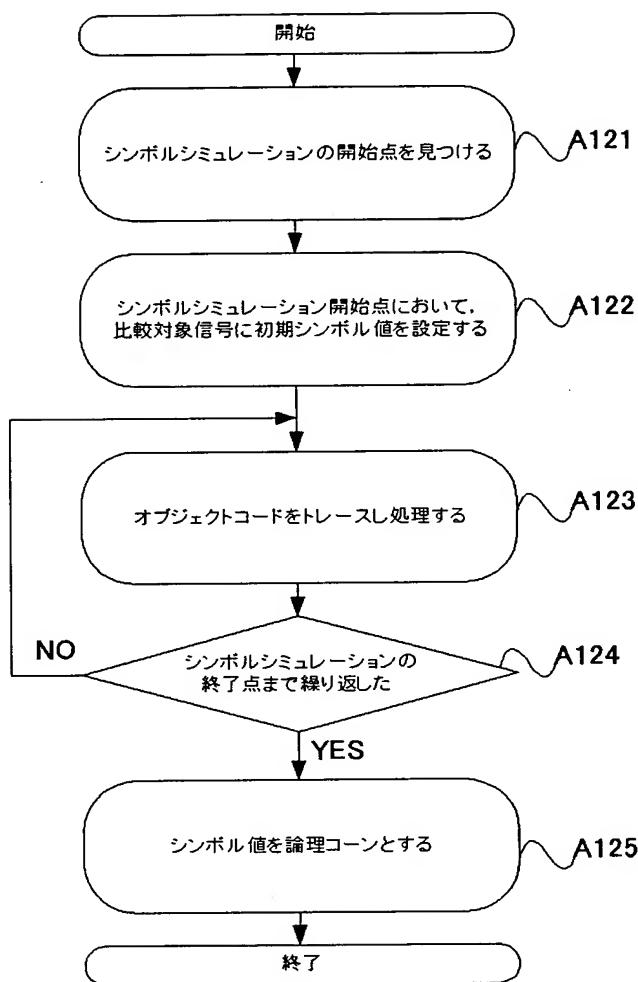
【図3】

【図3】



【図4】

【図4】



【図5】

【図5】

```

int a, b, c ; .....(1)
int in0, out0 ; .....(2)
#ifndef C .....(3)
main() { .....(4)
    while(1) { .....(5)
        scanf("%d", &in0) ; .....(6)
        addition(); .....(7)
        printf("%d\n", out0) ; .....(8)
    } .....(9)
#endif .....(10)

/* to behavioral Synthesis */ .....(11)
void addition() .....(12)
{
    b = a + b ; .....(13)
    a = in0 ; .....(14)
    out0 = b ; .....(15)
    return ; .....(16)
} .....(17)
.....(18)

```

12

【図6】

【図6】

```

module addition(in0, out0, CLOCK) ; .....(1)
input [31:0] in0 ; .....(2)
output [31:0] out0 ; .....(3)
input CLOCK .....(4)
reg[31:0] RG01; .....(5)
reg[31:0] RG02; .....(6)

assign out0 = RG02 ; .....(7)

always @ ( posedge CLOCK ) .....(8)
begin .....(9)
    RG01 <= in0 ; .....(10)
    RG02 <= RG01 + RG02 ; .....(11)
end .....(12)
endmodule .....(13)

```

14

【図7】

【図7】

15

```

addition: .....(1)
    movl a,%eax .....(2)
    addl %eax,b .....(3)
    movl in0,%eax .....(4)
    movl %eax,a .....(5)
    movl b,%eax .....(6)
    movl %eax,out0 .....(7)

```

【図8】

【図8】 対応情報の例（一部）

C言語記述上の信号	HD L記述上の信号
in0	in0
out0	out0
a	RG01
b	RG02

13

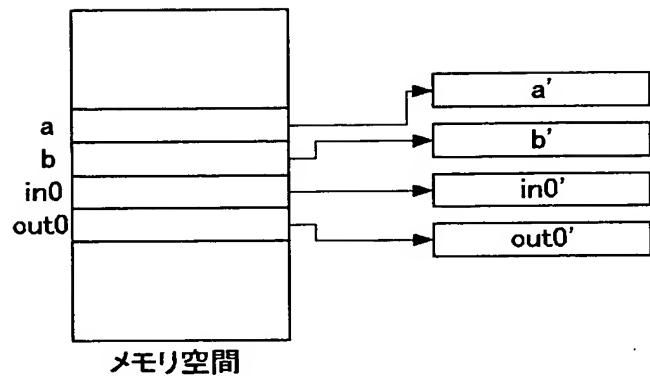
【図9】

【図9】 コンパイル情報の例（一部）

C言語記述上の信号	オブジェクトコード上の記憶領域
in0	in0
out0	out0
a	a
b	b

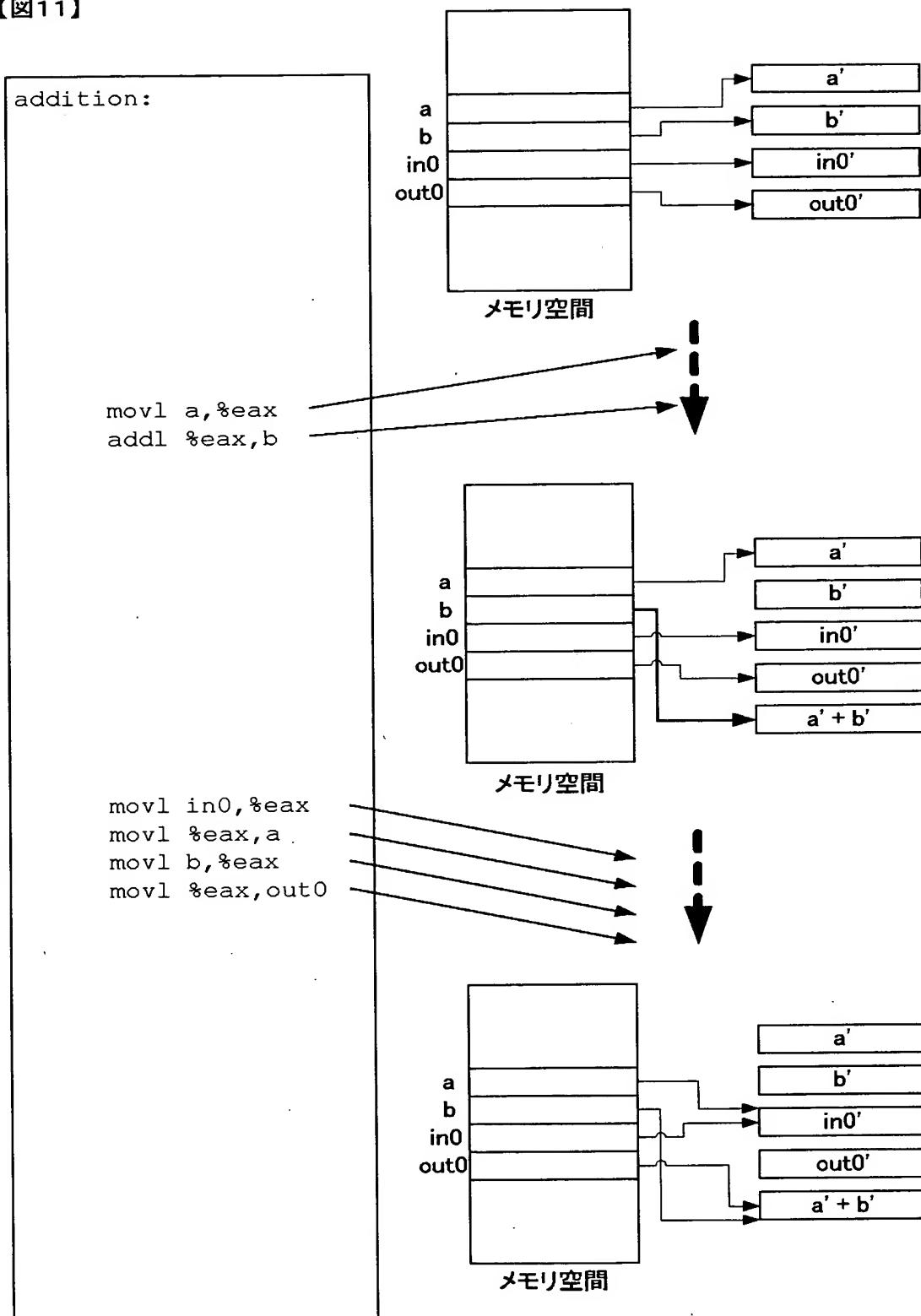
【図10】

【図10】



【図11】

【図11】



## 【図12】

【図12】 プログラム記述の論理コーンの例

変数	論理コーン
a	in0'
b	a' +b'
in0	in0'
out0	a' +b'

## 【図13】

【図13】 HDL記述の論理コーンの例

変数	論理コーン
RG01	in0'
RG02	RG01' +RG02'
in0	in0'
out0	RG01' +RG02'

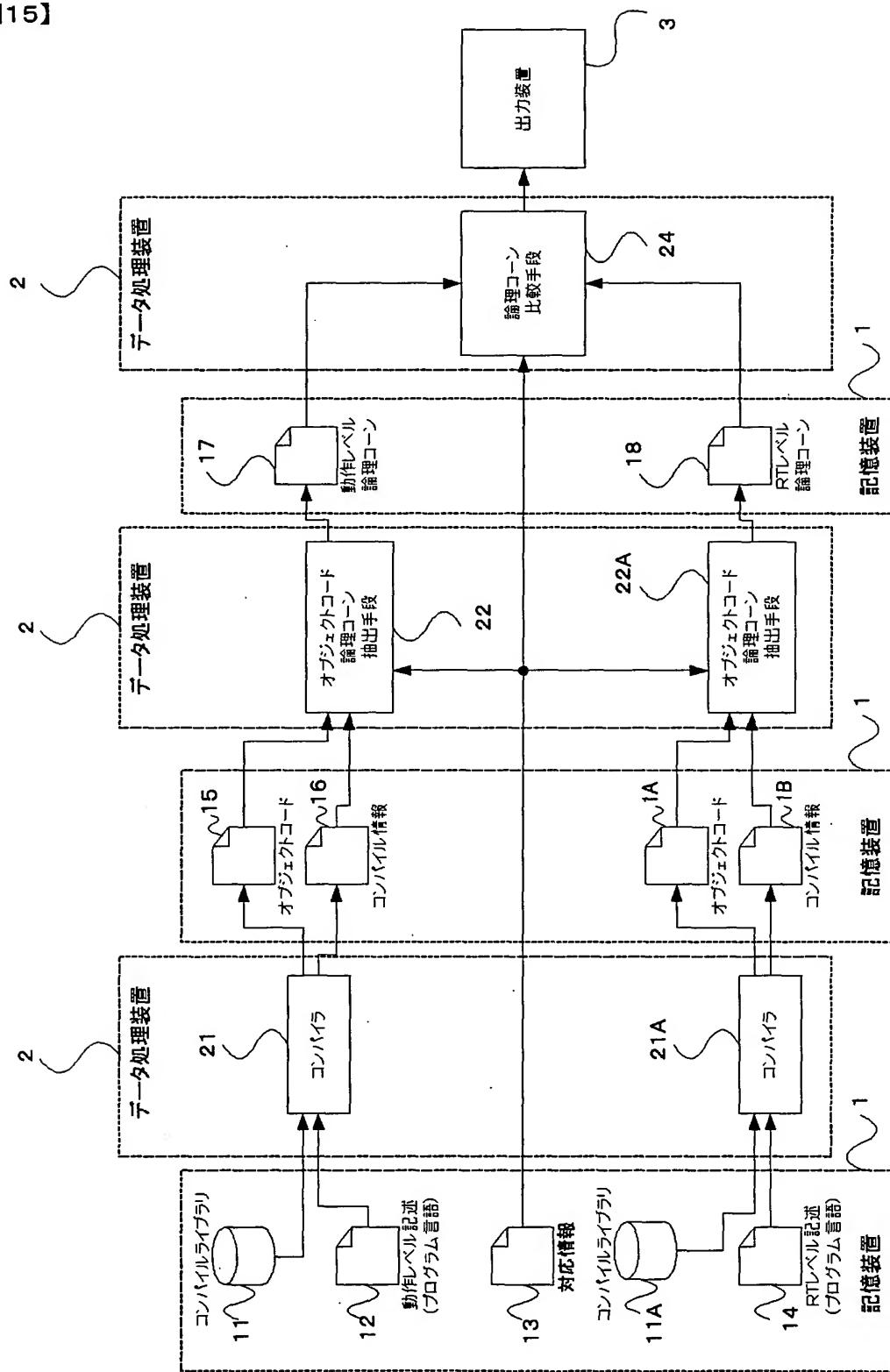
## 【図14】

【図14】 オブジェクトコード上の変数とHDL上の信号の対応

オブジェクトコード上の変数	HDL上の信号
a	RG01
b	RG02
in0	in0
out0	out0

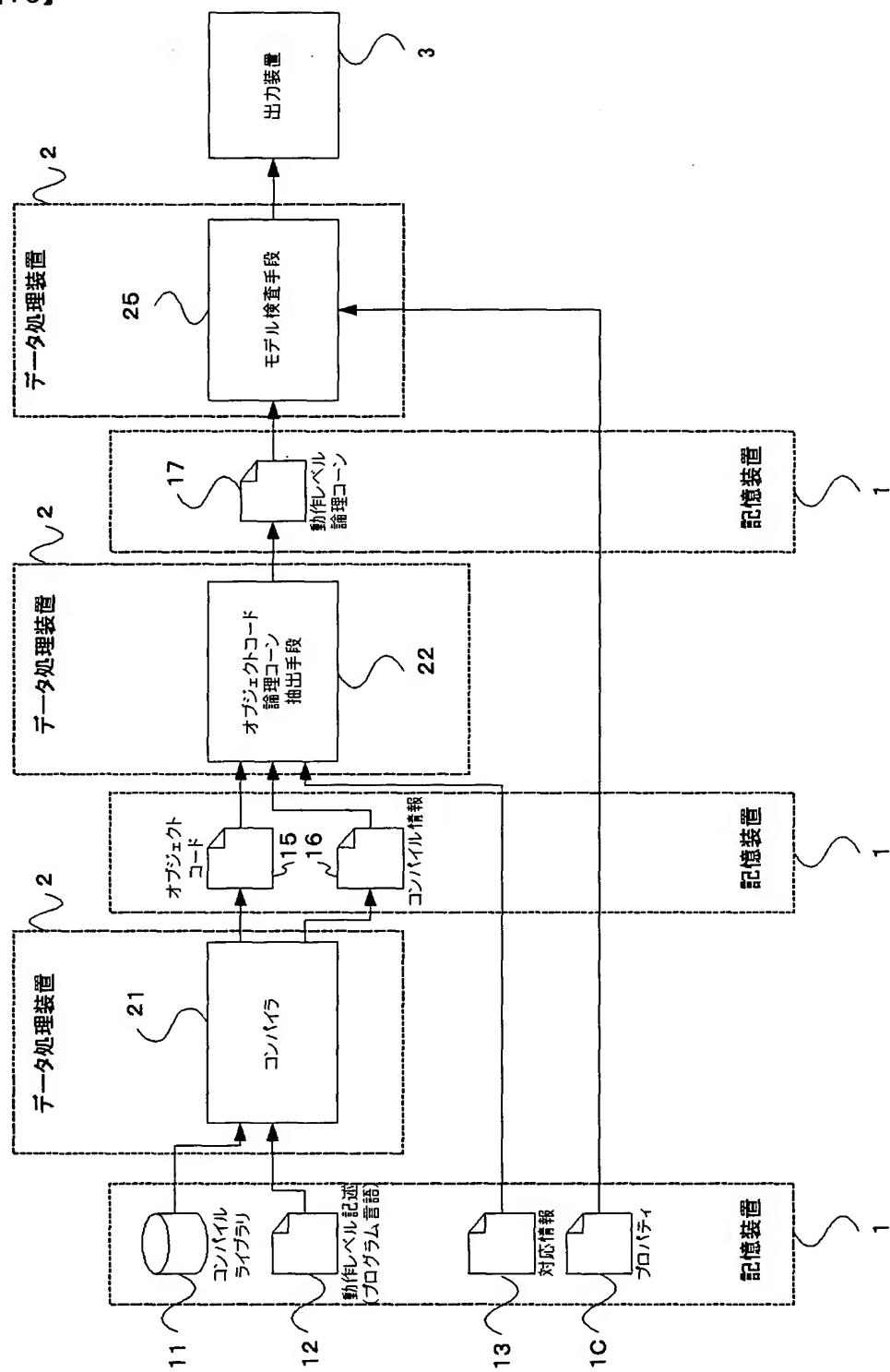
【図15】

【図15】



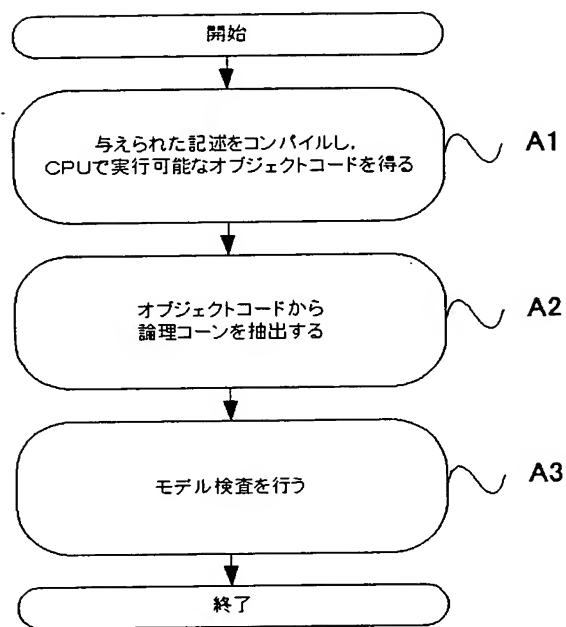
【図16】

【図16】



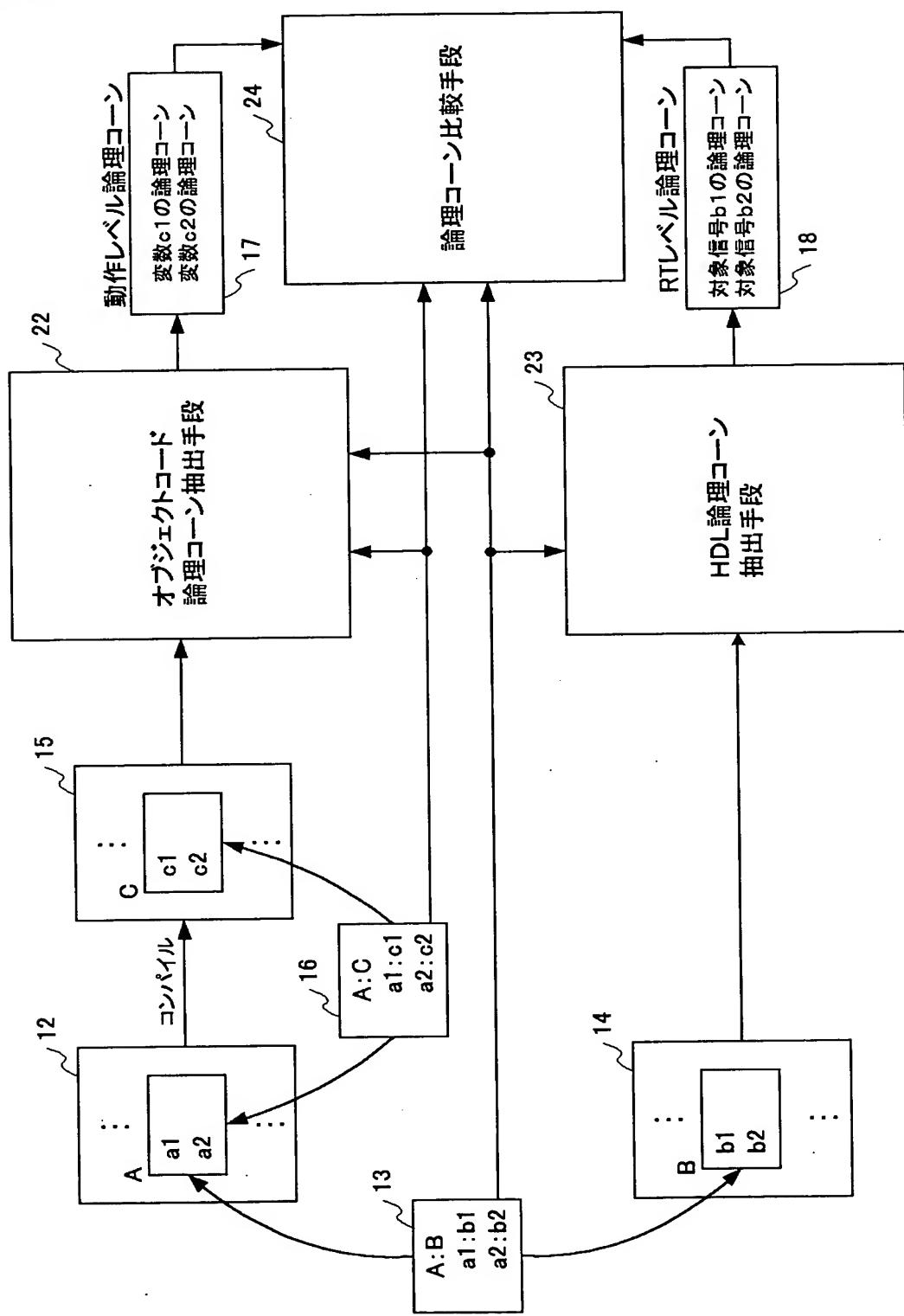
【図17】

【図17】



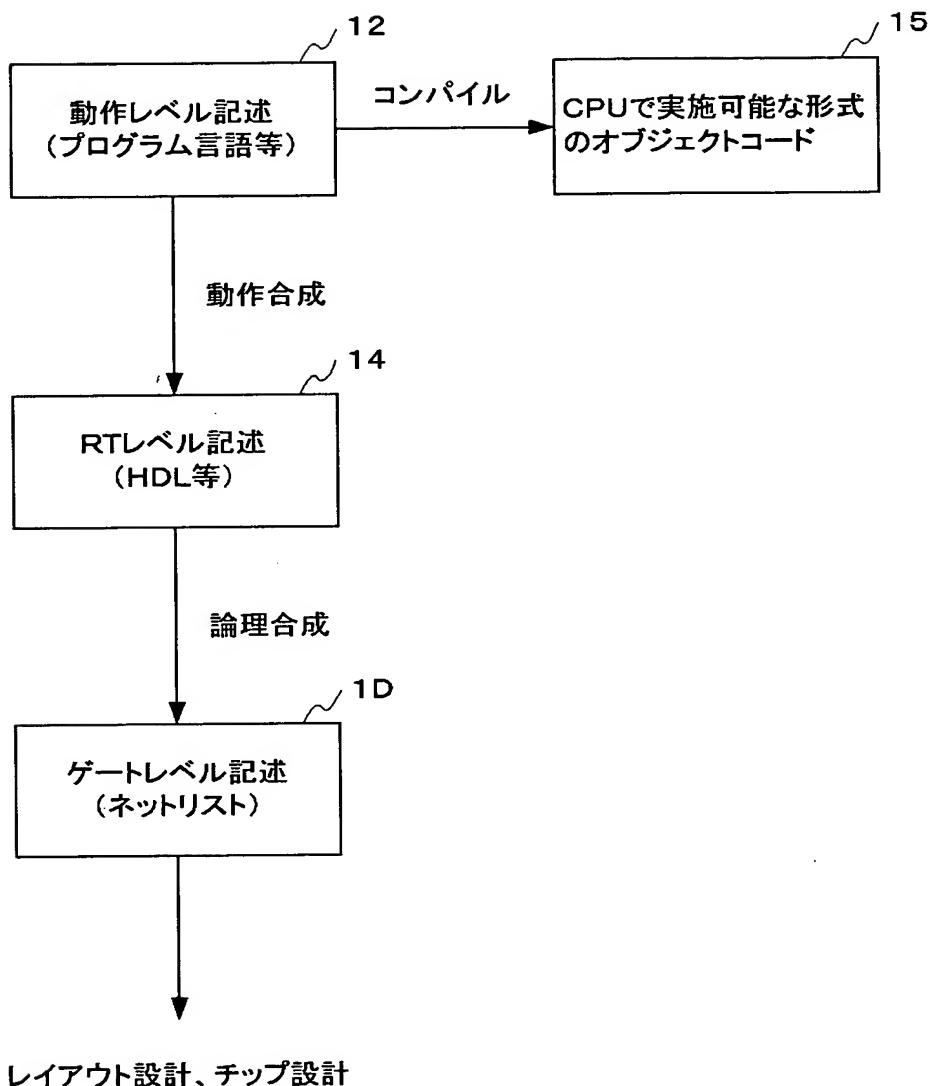
【図18】

【図18】



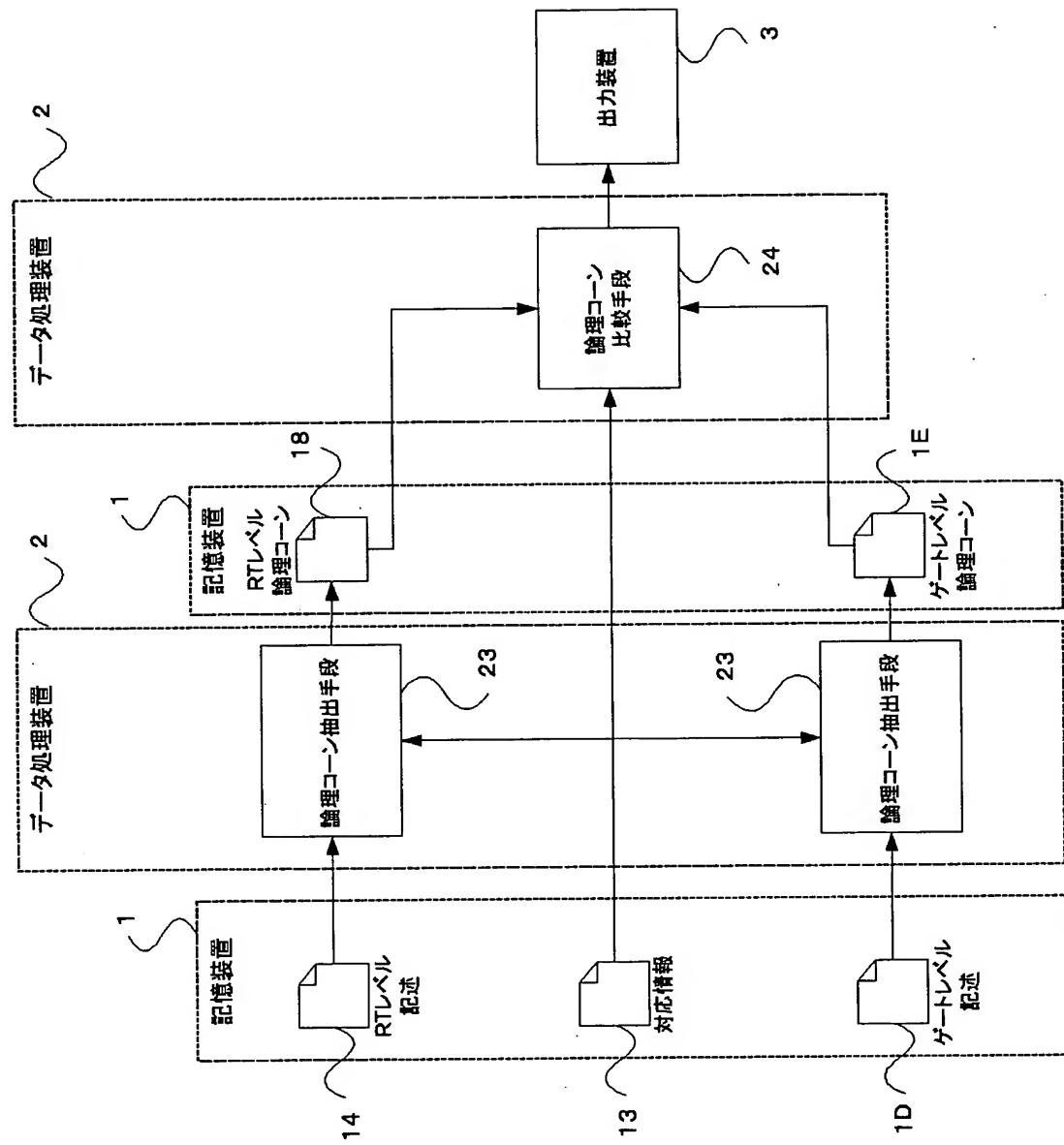
【図19】

【図19】



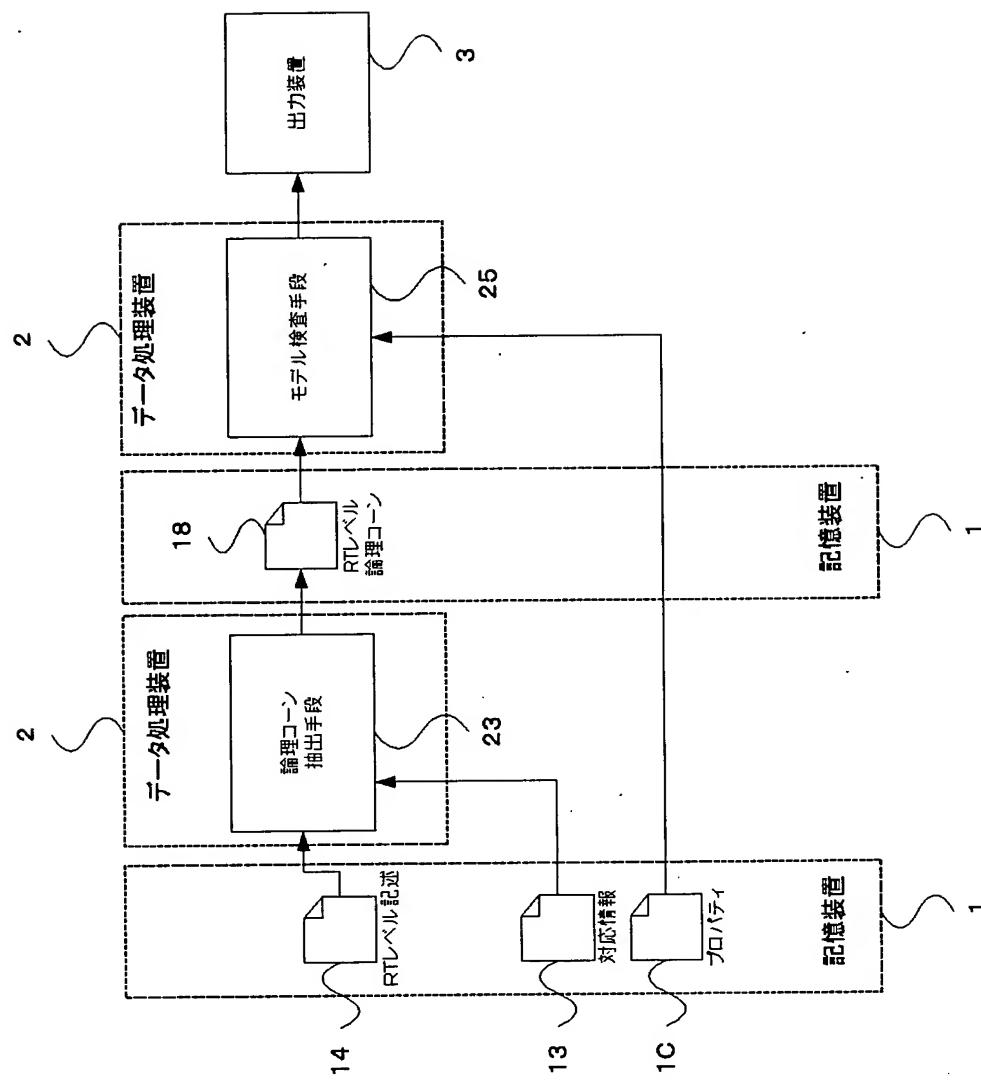
【図20】

【図20】



【図21】

【図21】



【書類名】 要約書

【要約】

【課題】 動作レベル記述をコンパイルしたオブジェクトコードと動作レベル記述から導出した RT レベル記述との等価性を検証する。

【解決手段】 対応情報 1 3 は、プログラム言語で記述された動作レベル記述 1 2 及びそれから動作合成した RT レベル記述 1 4 における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する。コンパイル情報 1 6 は、動作レベル記述 1 2 とオブジェクトコード 1 5 とのマッピング情報を含む。論理コーン抽出手段 2 3 は、対応情報 1 3 及びコンパイル情報 1 6 を参照して、オブジェクトコード 1 5 から、シンボルシミュレーションを行って、論理コーン 1 7 を抽出する。論理コーン抽出手段 2 3 は、 RT レベル記述 1 4 から論理コーン 1 8 を抽出する。論理コーン比較手段 2 4 は、論理コーン 1 7 と論理コーン 1 8 どうしの等価性を検証する。

【選択図】 図 1

出願人履歴情報

識別番号 [000004237]

1. 変更年月日 1990年 8月29日

[変更理由] 新規登録

住 所 東京都港区芝五丁目7番1号

氏 名 日本電気株式会社